(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 September 2005 (09.09.2005)

PCT

(10) International Publication Number
WO 2005/083926 A1

(51) International Patent Classification[7]: H04L 9/00, 9/32

(21) International Application Number:
PCT/US2005/006245

(22) International Filing Date: 25 February 2005 (25.02.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/547,502 25 February 2004 (25.02.2004) US

(71) Applicant *(for all designated States except US)*: THE TRUSTEES OF COLMBIA UNIVERSITY OF THE CITY OF NEW YORK [US/US]; 110 Low Memorial Library, 535 West 116th Street, New York, NY 10027 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: DIAMENT, Theodore [US/US]; 263 West End Avenue, #13C, New York, NY 10023 (US). KEROMYTIS, Angelos, D. [GR/US]; 423 West 120th Street, #102, New York, NY 10027 (US). YUNG, Marcel Mordechay [IL/US]; 401 East 80th Street, New York, NY 10021 (US). HOMIN, K., Lee [US/US]; 577 Bergen Street, #2, Brooklyn, NY 11238 (US).

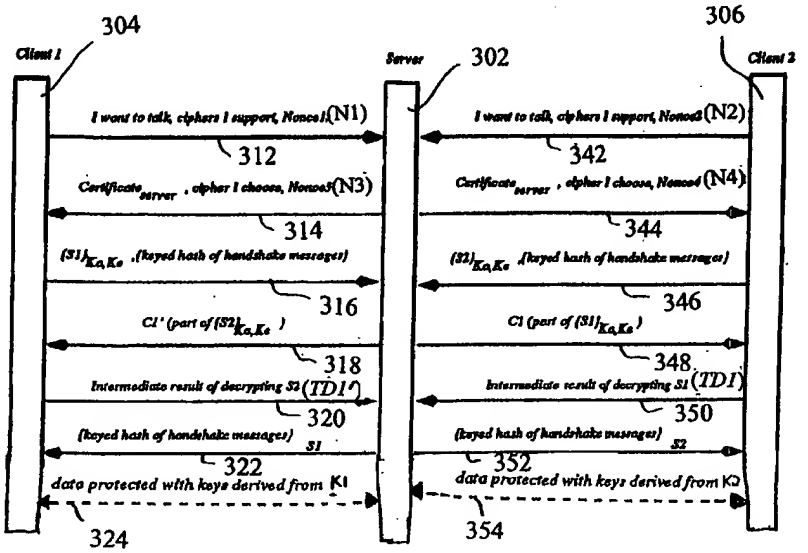
(74) Agent: KANABE, George, L.; Wilmer Cutler Pickering Hale and Dorr LLP, 399 Park Avenue, New York, NY 10022 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: COMPUTER-IMPLEMENTED METHODS AND SYSTEMS FOR GENERATING, SOLVING, AND/OR USING USEFUL SECURITY PUZZLES



(57) Abstract: A method for mitigating depletion of resources involving communication between a server (302) and two clients (304, 306).

WO 2005/083926 A1

FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

COMPUTER-IMPLEMENTED METHODS AND SYSTEMS FOR GENERATING, SOLVING, AND/OR USING USEFUL SECURITY PUZZLES

Cross-Reference to Related Application

[0001] This application claims the benefit under 35 U.S.C. §119(e) of United States Provisional Patent Application No. 60/547,502, filed February 25, 2004, which is hereby incorporated by reference herein in its entirety.

Field of the Invention

[0002] The present invention relates generally to the field of computer network security. More particularly, this invention relates to the use of useful security puzzles by a computer network or server to protect against resource-depletion attacks.

Background of the Invention

[0003] Computer viruses, worms, trojans, hackers, key recovery attacks, malicious executables, probes, etc. are a constant threat to the operation of computer servers that are connected to public computer networks (such as the Internet) and/or private networks (such as corporate computer networks). One type of threat that is of particular relevance to the present invention and is now explained is denial of service (DoS) attacks, or resource-depletion attacks.

[0004] In a typical client-server environment, for example, once a connection request from a client has been accepted, a channel is opened between the client and the server for data communications. For example, a channel may be opened between a computer shopper (client) and an online retail store server following a connection request by the computer shopper. In some instances, however, the client requesting the open channel for data communications can disrupt the operations of the server, either intentionally (as with DoS attacks) or unintentionally, by generating a large number of connection requests within a relatively short length of time, and thus unduly burdening the server's resources. A similar result can occur, for example, when one or more clients generate a large number of other types of requests that involve the use of one or more resources of the server. As a result, other computer users may be deprived of the resources they would normally expect to have access to from the server. For example, computer users may experience loss of e-mail

services being provided through the server, or, in the case of an online merchant or other type of Web site, computer users may be unable to access certain (or all) content on the Web site.

[0005] One approach to defending a network device, such as a server, against resource-depletion attacks, whether intentional or not, has been to rate limit remote clients by forcing them to solve one or more computational problems or puzzles before allowing a connection to be established with the server or allowing one or more resources of the server to be otherwise used. When in use by a server, these problems require clients to perform some computationally expensive computation (thus at least temporarily consuming some resources of the clients) before access (e.g., a connection) to the server is granted, or a resource of the server is allowed to be used. The impact of using such problems or puzzles on legitimate clients that issue only a few requests per time unit is generally modest. However, attackers seeking to exhaust the resources of a server by issuing large numbers of concurrent requests will need to perform considerable amounts of computations, making such attacks difficult to mount.

[0006] Typically, computation problems or puzzles used in defending against resource-depletion attacks as described above use a cryptographic one-way function (OWF) to construct a trapdoor function. In this case, a client presented with the result of such a OWF must use brute force to exhaustively search through the space of potential inputs (applying the OWF to each value) in order to determine the correct value. However, knowledge of the input to the OWF (which the client is forced to determine) allows the server to efficiently compute the result, making the "solution" provided by the client easily verifiable. For example, a server can ask the client a question such as "which 32-bit number, when supplied as the input to the Secure Hash Algorithm (SHA1) OWF, results in the value Oxdeadbeef?" The server can pick the input at random, and vary its size to reflect the computational resources of clients and attackers.

[0007] While requiring clients to solve computational problems or puzzles is at times an effective way to combat resource-depletion attacks, solving such problems has generally represented "useless" computation, given that no other purpose is served aside from rate-limiting requests.

[0008] Therefore, it would be beneficial to provide methods and systems for generating, solving, and/or using problems (e.g., useful security puzzles) that both protect against resource-depletion attacks and provide a useful service to the servers employing them.

Summary of the Invention

[0009] In accordance with the present invention, methods and systems are provided for generating, solving, and/or using problems (e.g., useful security puzzles) that can be used to protect against resource-depletion attacks, and whose solutions are useful to the network devices (e.g., servers) employing them. For example, in the security context, these problems or puzzles can be used by servers to protect against resource-depletion attacks and to offload at least some of the cryptographic overhead required by the servers for secure cryptographic key establishment.

[0010] In one embodiment, the invention provides a method for solving a problem using network devices in a computer network, where the method includes receiving, by a first network device, a first problem provided by a second network device, providing a second problem, by the first network device, to a third network device, wherein the second problem is based at least in part on the first problem, receiving a solution to the second problem by the first network device, and solving the first problem, by the first network device, using the received solution to the second problem.

[0011] In a second embodiment, the invention provides a first network device in a computer network that receives a first problem from a second network device, that provides a second problem that is based at least in part on the first problem to a third network device, that receives a solution to the second problem, and that solves the first problem using the received solution to the second problem.

[0012] In a third embodiment, the invention provides an article of manufacture that includes a computer usable medium having computer readable program code means embodied therein for solving a problem, where the computer readable program code means in the article of manufacture includes computer readable program code means for causing a first network device to receive a first problem from a second network device, computer readable program code means for causing the first network device to provide a second problem to a third network device, where the second problem is based at least in part on the first problem, computer readable program code means for causing the solution to the second problem to be

received by the first network device, and computer readable program code means for causing the first network device to use the received solution to the second problem to solve the first problem.

Brief Description of the Drawings

[0013] Additional embodiments of the invention, its nature and various advantages, will be more apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which like reference characters refer to like parts throughout, and in which:

[0014] FIG. 1 is a simplified illustration of a communications system in which the principles of the present invention may be implemented in accordance with at least one embodiment of the present invention;

[0015] FIG. 2 is a simplified illustration showing various steps involved in the creation of a communications link between a client and a server using the Transport Layer Security protocol;

[0016] FIG. 3 is a simplified illustration showing various steps involved in the creation of a communications link between a server and two clients using decryption-oriented puzzles in accordance with at least one embodiment of the present invention;

[0017] FIG. 4 is a simplified illustration showing various steps performed in connection with computing part of the trapdoor that is needed to generate decryption-oriented puzzles in accordance with at least one embodiment of the present invention; and

[0018] FIG. 5 is a simplified illustration showing various steps performed in connection with the encryption and decryption of decryption-oriented puzzles in accordance with at least one embodiment of the present invention.

Detailed Description of the Invention

[0019] In the following description, numerous specific details are set forth regarding the methods and systems of the present invention and the environment in which such methods and systems method may operate, etc., in order to provide a thorough understanding of the present invention. It will be apparent to one skilled in the art, however, that the present invention may be practiced without such specific details, and that certain features which are

well known in the art are not described in great detail in order to avoid complication of the subject matter of the present invention. Moreover, it will be understood that the examples provided below are exemplary, and that it is contemplated that there are other methods and systems that are within the scope of the present invention.

[0020] Generally speaking, the present invention is directed to methods and systems for generating, solving, and/or using problems (e.g., useful security puzzles) that both provide a useful service to the servers employing them, and protect the servers protection against resource-depletion attacks from clients. It is noted that the terms "problems" and "puzzles" are used interchangeably herein. For example, as described below, a problem or puzzle as described below can be a ciphertext (e.g., plaintext that has been encrypted using some form of encryption algorithm) that requires some form of computation to decrypt, and which may subsequently be used by a server to generate a secure cryptographic key. However, the invention is not limited by the particular types of problems or puzzles being used. Moreover, as used herein, the terms "client" and "server" are used generally to differentiate end points in a network connection. It will be understood, however, that clients and servers according to the invention can be any suitable type of computing or network device that is capable of communicating across a network. For example, the clients and servers discussed herein can be a laptop or other personal computer, a mobile (cellular) telephone, a personal digital assistant (PDA), a mainframe server, and so on. Additionally, for example, in accordance with the invention, a "client" may be a program that sends a request for information from a "server." Moreover, it will be understood that a server in one instance may be a client in another (and vice versa).

[0021] In accordance with the principles of the present invention, useful security puzzles have some (or all) of the following characteristics. First, according to various embodiments, the security puzzles represent at least a moderate computational task, thus assuring a certain slow-down of the accessing parties (clients). Second, according to various embodiments, it is computationally efficient for the server to verify the results of the puzzles. That is, the solutions to the useful security puzzles arrived at by clients are at least somewhat (and potentially much) easier to check than to compute. Third, according to various embodiments, the computations associated with the useful security puzzles are useful to the server. For example, solving a useful security puzzle can include performing a computational task that makes the server's computations relating to secure cryptographic key establishment more

efficient. Fourth, according to various embodiments, solving the useful security puzzle does not depend on any particular client. In other words, if a given puzzle is not actually solved by the first client it is given to, the server is still able to solve it (or give the puzzle to another client to solve). Fifth, the useful security puzzle should be such that, even when a client determines the solution to a puzzle, the client does not learn any long-term cryptographic keys or other secret information of the server.

[0022] An example of useful security puzzles in accordance with the principles of the present invention is now provided in the context of security. Referring to communications system 100 shown in FIG. 1, consider a network device (e.g., Web server) 102 with which another network device (e.g., a client) 104 seeks to initiate a channel, or communications link 106 over a network, such as the Internet 108. In initiating communications link 106, client 104 attempts to negotiate (using a handshaking session) how information should be securely transmitted. For example, client 104 may seek to initiate communications link 106 using a session-security protocol such as the Transport Layer Security protocol (TLS) described in RFC 2246, which is incorporated by reference herein in its entirety. A brief description of the manner in which a client connects to a server according to TLS is now provided immediately below with reference to FIG. 2. It is noted that, although TLS with a Rivest, Shamir, and Adelman (RSA) key exchange is described, other types of key exchange are also possible. For example, a Diffie-Hellman key exchange can also be used.

[0023] FIG. 2 is a simplified illustration showing various steps involved in the initiation of communications link 106 of FIG. 1 (and subsequent protection of data transmissions using the derives session keys) when TLS (with RSA key exchange) is being used. In step 202, client 104 extends its "hand" by informing server 102 that it wishes to "talk" (communicate) using TLS. At this time, client 104 provides various information about itself (e.g., the ciphers it supports) to server 102, and optionally also provides "nonce" *Nc*. It will be understood that, as used herein, the term "nonce" refers to a randomly generated value that can be used to detect replay attacks.

[0024] In response, in step 204, server 102 extends its "hand" with a reply containing a certificate that can be used in authenticating it, various information about itself (e.g., the selected cipher), and a nonce *Ns*. Using the received certificate and other information, client 104 authenticates server 102. Although not shown in FIG. 2, server 102 could also ask for a

certificate from client 104 when it wishes to authenticate client 104 (e.g., prior to an online financial transaction).

[0025] If the client 104 is able to authenticate the server 102 in step 204, in step 206, client 104 generates a randomly-chosen secret message *S*, encrypts it with the server's public key (obtained, e.g., from the certificate sent by the server 102 in step 204) using the RSA algorithm, and sends the encrypted secret message *S* to server 102. Once it is received, server 102 decrypts the encrypted secret message *S*.

[0026] As described above, when using an RSA key exchange mechanism, client 104 selects the secret message *S* without any input from server 102. However, in step 208, an additional hashing step is used whereby server 102 can supply input in the derivation of the master secret from the secret message *S*.

[0027] Finally, in step 210, both client 104 and server 102 use the secret message *S*, also called a "pre-master secret," to derive a "master secret" *K* using additional information, such as *Nc*, *Ns*, and information resulting from step 208 described above. Using the derived master secret *K*, both client 104 and server 102 are able to generate session keys to be used for encrypting and decrypting various communications between the two.

[0028] In accordance with the principles of the present invention, the use of useful security puzzles as a substitute for the RSA encryption described above is now described. It will be understood that these puzzles, which in this context are referred to herein as "decryption-oriented puzzles," can be used by a server (such as server 102) not only to protect against resource-depletion attacks by one or more clients, but also to allow the server to offload much of the cryptographic overhead required for secure cryptographic key establishment.

[0029] FIG. 3 is a simplified illustration showing various steps involved in the creation of communications links when decryption-oriented puzzles are used in place of RSA encryption in connection with a TLS-like protocol. Server 302 shown in FIG. 3 has a permanent public key *Ke*, and a periodically generated auxiliary public key *Ka*. According to various embodiments, both public keys *Ke* and *Ka* are arbitrarily chosen and, given a particular permanent public key *Ke*, any choice for auxiliary public key *Ka* can be used. In addition, server 306 has a permanent private key *Pe*, and a periodically generated auxiliary private key *Pa*. It is noted that, while only a first client 304 and a second client 306 are shown as

communicating with server 306, this is for simplicity only, and the invention is not limited by the number of clients.

[0030] In step 312, similar to step 202 described above in connection with FIG. 2, client 304 extends its "hand" by informing server 302 that it wishes to "talk" (communicate) using the TLS-like protocol. At this time, client 304 provides various information about itself (e.g., the ciphers it supports) to server 302, and optionally also provides nonce *N1*.

[0031] In response, in step 314, server 302 extends its "hand" with a reply containing a certificate that can be used in authenticating it, various information about itself (e.g., the selected cipher), and a nonce *N3*. Using the received certificate and other information, client 304 authenticates server 302. Moreover, although not shown in FIG. 3, server 302 could also ask for a certificate from client 304 when it wishes to authenticate client 304 (e.g., prior to an online financial transaction).

[0032] If the client 304 is able to authenticate the server 302 in step 314, in step 316, client 304 generates a randomly-chosen secret message *S1* (similar to the "pre-master secret *S*" described above), and encrypts it into a decryption-oriented puzzle (using the principles described below) using both the permanent public key *Ke* and auxiliary public key *Ka* of server 302 (obtained, e.g., from the certificate sent by the server 102 in step 204). The decryption-oriented puzzle based on secret message *S1*, which is provided to server 302, is a ciphertext that has two portions (*C1*, *C2*). As explained below, the decryption-oriented puzzle is generated such that, given ciphertext (*C1*, *C2*), the secret message *S1* can be recovered using either the server's permanent private key *Pe*, or the server's auxiliary private key *Pa*. However, given only *C1* and one of the private keys *Pe* and *Pa*, only an intermediate value, *TD1*, can be obtained by using a "trapdoor recovery algorithm."

[0033] Also in step 316, client 304 sends the generated decryption-oriented puzzle, including both ciphertext portions (*C1*, *C2*), to server 302. Once the puzzle is received by server 302, if it does not wish to offload any computation on another client (e.g., because it is lightly loaded), server 302 may decrypt the puzzle itself (thus recovering secret message *S1*) using either permanent private key *Pa* or auxiliary private key *Pe*. Then, using the recovered secret message *S1*, along with additional information (e.g., *N1* and *N3*), server 302 is able to derive the master secret *K1*. Otherwise, in order to offload some of the cryptographic overhead required to recover the original secret message *S1*, in step 348, server 302 forwards only

ciphertext portion *C1* to another accessing client 306 after steps 342-346 (which are similar to steps 312-316, but involve client 306 rather than client 304) have been performed. Moreover, if the auxiliary private key *Pa* of server 302 was not previously sent to client 306 (e.g., with the certificate at step 344), then in step 348, server 302 also provides this key to client 306 so that client 306 can solve puzzle *C1* and obtain trapdoor *TD1*. It is noted that, on a busy server, such as a popular e-commerce Web site, there will be a constant stream of new clients such as client 306 connecting to server 302 to which *C1* can be forwarded. Similarly, as explained below, client 304 may receive another *C1'* that was produced by another client (e.g., client 306) connecting to server 302.

[0034] After it has received *C1* and auxiliary private key *Pa*, in step 350, client 306 uses *Pa* to produce the intermediate value *TD1*, and sends this result back to server 302 as proof of work done. If server 302 verifies the solution (*TD1*) produced by client 306, it will allow the connection process associated with client 306 to proceed. In accordance with the principles of the present invention, as explained further below, given *TD1* and ciphertext portions (*C1*, *C2*), the secret message *S1* can be efficiently recovered by server 302 using a "message recovery algorithm" that is substantially more efficient than the trapdoor recovery algorithm used by client 306 to obtain *TD1*. In addition, according to various embodiments, the message recovery algorithm being used by server 302 to recover secret message *S1* is able to efficiently detect an incorrect *TD1* as explained below. It is be noted that, if the recovery of secret message *S1* in step 350 fails (e.g., because client 306 did not solve *C1* to obtain the correct *TD1*, or did not provide *TD1* to the server for another reason), server 302 can still solve puzzle *C1* itself in order to obtain *TD1* and subsequently recover secret message *S1*. Alternatively, server 302 can try to employ another accessing client to recover *TD1*.

[0035] In step 322, an additional hashing step is used whereby server 302 can supply input in the derivation of the master secret from the secret message *S1*. Then, in step 324, both client 304 and server 302 use the secret message *S1* to derive a "master secret" *K1* using additional information, such as *N1*, *N3*, and information resulting from step 322 described above. Using the derived master secret *K1*, both client 304 and server 302 are able to generate session keys to be used for encrypting and decrypting various communications between the two.

[0036] It is noted that, when client 304 is provided *C1'* that was produced by another client (e.g., client 306) in step 318, it must return the solved *TD1'* associated with that *C1'* in order

for server 302 to allow the connection process associated with client 304 to proceed (i.e., the correct *TD1'* must be returned before steps 322 and 324 are carried out). Moreover, once *TD1'* is received and verified by server 302, steps 350 and 352 (relating to client 306) may follow whereby client 306 and server 302 are both able to generate sessions keys using derived master secret key *K2*. It will be understood that steps 318-324 are similar to steps 348-354, but involve client 304 rather than client 302.

[0037] Even when clients 304 and 306 help server 302 recover secret messages *S2* and *S1*, respectively, in the manner described above, preferably neither client 304 nor client 306 has learned anything about these secret values they helped to decrypt. This is because, as mentioned above and explained in greater detail further below, even when given *C1* and one of the private keys *Pe* and *Pa*, a client (e.g., client 304 or client 306) is not able to determine or predict the secret message that was used to obtain ciphertext (*C1*, *C2*). Moreover, knowledge of the private auxiliary key(s) *Pa* provided to clients 304 and 306 (the same key may be used, but this is not required) does not affect security, as private auxiliary key *Pa* will generally be relatively short lived.

[0038] It will be appreciated that the decryption-oriented puzzles that are used, e.g., in the manner described above with reference to FIG. 3 adhere to each of the five characteristics of useful security puzzles described above. For example, in order to protect the server employing them, these decryption-oriented puzzles represent at least a moderate computational task (as necessary to assure a certain slow-down) to accessing clients (e.g., clients 304 and 306). It will be understood that the complexity of the puzzles, the generation of which is described below, can be varied in accordance with the desired level of protection (slow-down of accessing clients) and various other factors.

[0039] In addition, these decryption-oriented puzzles are "useful" to server 302, because the computation associated with the decryption-oriented puzzles can be used for more than rate limiting connection requests by clients (e.g., to assist server 302 in solving a secret message generated by another client). Thus, by allowing server 302 to offload much (or at least some) of the cryptographic overhead required for secure cryptographic key establishment, the computational processing that would otherwise be required by server 302 can be significantly reduced.

[0040] Moreover, as mentioned above, server 302 is able to efficiently verify that the solutions (*TD1s*) generated by clients for the decryption-oriented puzzles are correct, because solutions to the puzzles can be checked with much less computation than is required to generate the solutions. Also, these decryption-oriented puzzles are such that the server can still solve them if the clients do not (although this required greater computation on the part of the server), and secrets of the server are not provided to the clients in order for the clients to solve the puzzles, and, similarly, the clients are not in possession of any secrets from the end results (i.e., the solutions).

[0041] Having described one of the potential uses for description-oriented puzzles above, the generation scheme for generating such decryption-oriented puzzles is now described in greater detail. First, it is noted that, according to various embodiments, the generation scheme can be based on any bilinear map between two groups which a public key encryption can be based on, using pairing defined on certain elliptic curves. Several different designs, or schemes, have been implemented in which pairings are used to construct cryptosystems. For example, one design is the three-party Diffie-Hellman (DH) key exchange that is discussed in "A one-round protocol for tripartite Diffie-Hellman," Antoine Joux, 2000, which is hereby incorporated by reference in its entirety. Another design relates to an identity-based encryption (IBE) scheme in which the public key is a user's identity and a key-generation authority assigns the users private keys. In this scheme, which is discussed in "Identity-based encryption from the Weil pairing," Boneh and Franklin, 2001, which is hereby incorporated by reference in its entirety, key-escrow is inherent, as the key-generation authority knows all the users' private keys. The capability of pairing-based cryptography was previously noted in "Evidence that XTR is more secure than supersingular elliptic curve cryptosystems," Eric Verheul, 2001, which is hereby incorporated by reference in its entirety. Moreover, as discussed in "Short signatures from the Eeil pairing," Boneh et al., 2001, pairings can be used to generate short signatures. Various other constructions have also been suggested to date.

[0042] The scheme provided in accordance with the present invention for generating decryption-oriented puzzles is based on the Tripartite Diffie-Hellman (TDH) algorithm, which is explained further below, in that the security of this scheme is based at least in part on the difficulty of the TDH problem (task). A brief description of TDH and related algorithms is now provided.

[0043] The TDH problem (tasks) is an extension of the following three problems (tasks) for a multiplicative group G. The first of these problems, referred to as the Discrete Logarithm (DL) problem, is defined as follows: given two groups elements g and h, find an integer n such that $h = g^n$, whenever such an integer exists.

[0044] The second of these problems, referred to as the Computational Diffie-Hellman (CDH) problem, is defined as follows: given three groups elements g, g^a, and g^b, where $a,b \in Z$ (and Z represents the set of integers), find an element h such that $h = g^{ab}$. In connection with the CDH problem, a CDH Parameter Generator is defined as a randomized algorithm that takes a security parameter k, and outputs the description of a group G for which the CDH problem is hard.

[0045] The third of these problems, referred to as the Decision Diffie-Hellman (DDH) problem, is defined as follows: given four groups elements g, g^a, g^b, and g^c, where $a, b, c \in Z$, decide whether or not $c = ab$ (modulo the order of g). Given these three problems, it is noted that the DDH problem is no harder than the CDH problem, and that the CDH problem is no harder than the DL problem.

[0046] As mentioned above, the scheme for generating decryption-oriented puzzles is based at least in part on the difficulty of the TDH problem, which is itself an extension of the above three problems. In particular, the TDH problem is defined as follows: given groups elements P, aP, bP, and cP in G_1, where $a, b, c \in Z$, find an element $g \in G_2$ such that $g = \langle P,P \rangle^{abc}$. In connection with the TDH problem, a TDH parameter generator is defined as a randomized algorithm that takes a security parameter k, and outputs the description of two groups G_1 and G_2, and the description of a non-degenerate bilinear map between the two groups for which the TDH problem is hard.

[0047] According to various embodiments of the invention, the scheme for generating decryption-oriented puzzles makes use of a non-degenerate pairing (i.e., a bilinear map between two groups G_1 and G_2). The pairing of two elements $P, Q \in G_1$ is denoted as $\langle P,Q \rangle \in G_2$. Due to the bilinearity condition, for all $P, Q \in G_1$ and $a, b \in Z$, the pair $\langle aP,bQ \rangle = \langle P,Q \rangle^{ab}$. Note that, according to various embodiments, the DL problem should be hard in G_2 so that the pairing is not easily invertible and the DL problem in G_1 is not easily solved. In accordance with various embodiments of the present invention, the Weil and/or

the Tate pairings defined over points on an elliptic curve defined over a finite field are chosen for such bilinear maps. An analysis of TDH wherein the Weil and Tate pairings are used as building blocks for cryptosystems is provided in "The Weil and Tate pairings as building blocks for public key cryptosystems," Antoine Joux, 2002, which is hereby incorporated by reference herein in its entirety. Several details of the bilinear mapping used in the scheme for generating decryption-oriented puzzles in accordance with the invention are discussed in the Appendix below.

[0048] In accordance with the principles of the present invention, a system is now presented that computes part of the trapdoor that is needed to generate decryption-oriented puzzles such as the ones used by server 302 in the manner described above with reference to FIG. 3. This partial system performs the steps shown in the flow chart of FIG. 4.

[0049] In step 402, groups G_1 and G_2 are chosen using the TDH parameter generator described above, along with a random element $P \in G_1$ and $y \in Z$. The server's permanent public key (*Ke*) is set to (P,yP), and the permanent private key (*Pe*) is set to y. A cryptographic hash function H: $G_2 \rightarrow \{0,1\}^n$ is also generated in this step.

[0050] In step 404, a random element $x \in Z$ is chosen. Using random element x, the auxiliary public key (*Ka*) is set to (P,xP) and the auxiliary private key (*Pa*) is set to as x.

[0051] Then, in step 406, a random element $r \in Z$ is chosen by the encryption algorithm, and ciphertext u_1 is computed where $u_1 = rP$.

[0052] In step 408, given ciphertext u_1, the first part of the decryption algorithm computes $\langle u_1,yP\rangle^x$. It is noted that there are many ways to compute the trapdoor because, due to bilinearity, $\langle u_1,yP\rangle^x = \langle rP,yP\rangle^x = \langle xP,yP\rangle^r = \langle P,P\rangle^{rxy}$.

[0053] In step 410, given a ciphertext C1 = u_1, the trapdoor recovery algorithm computes $\langle u_1,xP\rangle^y$. It is noted that $\langle u_1,xP\rangle^y = \langle rP,xP\rangle^y = \langle xP,yP\rangle^r = \langle P,P\rangle^{rxy}$.

[0054] It is noted that the trapdoor encryption described above is a one-way function (analogous to the DH computation over a finite field), as it is computable using either the random r, the private key y, or the auxiliary private key x, but is otherwise hard to compute.

[0055] Based on this hardness and based on the REACT transform (employing strong cryptographic hash functions that behave like a random oracle) that is discussed in "REACT:

rapid enhanced-security asymmetric cryptosystem transform," Okamoto and Pointcheval, 2002, which is hereby incorporated by reference herein in its entirety, it is possible to build the entire system as follows so that it is chosen ciphertext secure (e.g., secure against chosen-ciphertext attacks, in which an attacker tries to determine information about a secret key by examining correlations between a series of ciphertexts and their respective decryptions) and has the desired puzzle properties to be classified as a useful puzzle. In particular, the REACT conversion is used to convert the one-way preliminary scheme applying the TDH into a chosen-ciphertext secure scheme. The message space in connection with this full system is $M = \{0,1\}^{n_1}$. In accordance with the principles of the present invention, this system performs the steps shown in the flow chart of FIG. 5, which are now described.

[0056] In step 502, groups G_1 and G_2 are chosen using a TDH parameter generator as described above, as are a random element $P \in G_1$ and $y \in Z$. The permanent public key (*Ke*) is set to (p,yP) and the permanent private key (*Pe*) is set to y. In addition, the following three cryptographic hash functions are generated: H: $G_2 \rightarrow \{0,1\}^{n_1}$, G: $\{0,1\}^{n_1} \rightarrow \{0,1\}^{n_1}$, and F: $\{0,1\}^{4n_1} \rightarrow \{0,1\}^{n_2}$.

[0057] In step 504, a random element $x \in Z$ is chosen. In addition, the public auxiliary key (*Ka*) is set to (P,xP) and the private auxiliary key (*Pa*) is set to x.

[0058] In step 506, the plaintext input of $m \in \{0,1\}^{n_1}$ (which can correspond to one of the "secret messages" described above in connection with FIG. 3), is encrypted. In particular, the encryption algorithm chooses a random element $r \in Z$ and a random element $p \in \{0,1\}^{n_1}$, and computes $u_1 = rP$, $u_2 = p \oplus H(\langle xP,yP\rangle^r)$, $u_3 = m \oplus G(p)$, and $u_4 = F(p,m,u_2,u_3)$. The resulting cyphertext is ($C1 = [u_1,u_2]$, $C2 = [u_3,u_4]$).

[0059] In step 508, given a ciphertext u_1, the first part of the decryption algorithm computes, $\langle u_1,yP\rangle^x$ and then $u_2 \oplus H(\langle u_1,yP\rangle^x) = p = TD1$.

[0060] In step 510, given ciphertext ($C1 = [u_1,u_2]$, $C2 = [u_3,u_4]$) and the trapdoor $TD1 = p$, the message recovery algorithm computes $G(p) \oplus u_3 = m$, and it checks that $u_4 = F(p,m,u_2,u_3)$. If it is determined in step 512 that u_4 is correct, in step 514, the algorithm outputs m. Otherwise, in step 516, it optionally outputs "Reject" to indicate that the client that provided *TD1* to it should not be provided access because it did not properly solve the *C1* portion of the ciphertext. In this case, in step 518, the server optionally computes *TD1* for itself from

u_1,u_2, and attempts to decrypt the message again in step 520 (this time, checking the integrity of the sender of the ciphertext ($C1 = [u_1,u_2]$, $C2 = [u_3,u_4]$). Although not shown, alternatively, a new client could be asked by the sender at this point to compute *TD1*. If it is determined in step 522 that u_4 is correct, then in step 514, the algorithm outputs *m*. Otherwise, the algorithm ends or other suitable steps are performed, as it is assumed that there is a problem with the ciphertext received by the server.

[0061] It will be understood that, while the trapdoor recovery (which results in *p*) gives only a random value, it involves the costly (computationally intensive) operation over the curve (the pairing). Accordingly, a server rate limits requests by clients by requiring such clients to compute *p* (*TD1*), as described above in connection with server 302 shown in FIG. 3. On the other hand, given *TD1*, the message recovery by the server involves only bit-wise XOR, and a check of a simple hash function (which is much less computationally intensive). Moreover, if a client is not able (or willing) to compute *p* (*TD1*), as can be determined from a check in REACT, the server is able to perform the computation itself. In light of the above, it will also be understood that the scheme is a chosen-ciphertext secure public-key encryption scheme if the TDH problem is assumed to be hard (one way). The clients, seeing only part of the ciphertext that recovers to a random value p, will have no idea what the message is that is being decrypted by the server.

[0062] Although the invention has been described and illustrated in the foregoing illustrative embodiments, it is understood that the present disclosure has been made only by way of example, and that numerous changes in the details of implementation of the invention can be made without departing from the scope of the invention. For example, while decryption-oriented puzzles for use in TLS-like protocols is described in detail above, the invention is not limited in this manner. Rather, it is contemplated that other types of useful security puzzles will be used in various other settings that do not involve creating a communications link or connection between two network devices.

[0063] Therefore, other embodiments, extensions, and modifications of the ideas presented above are comprehended and should be within the reach of one versed in the art upon reviewing the present disclosure. Accordingly, the scope of the present invention in its various aspects should not be limited by the examples presented above. The individual aspects of the present invention, and the entirety of the invention should be regarded so as to

allow for such design modifications and future developments within the scope of the present disclosure. The present invention is limited only by the claims which follow

[0064] It is to be understood that the invention is not limited in its application to the details of construction and to the arrangements of the components set forth in the following description or illustrated in the drawings. The invention is capable of other embodiments and of being practiced and carried out in various ways. Also, it is to be understood that the phraseology and terminology employed herein are for the purpose of description and should not be regarded as limiting.

[0065] As such, those skilled in the art will appreciate that the conception, upon which this disclosure is based, may readily be utilized as a basis for the designing of other methods and systems for carrying out the several purposes of the present invention, which is limited only by the claims that follow. It is important, therefore, that the claims be regarded as including such equivalent constructions insofar as they do not depart from the spirit and scope of the present invention.

[0066] The following references are incorporated by reference herein in their entireties:

A. Back. Hashcash - A Denial of Service Counter-Measure. http://www.cypherspace.org/hashcash/hashcash.pdf, August 2002.

D. Boneh and M. Naor. Timed Commitments (Extended Abstract). In Proceedings of CRYPTO, pages 236-254, August 2000.

Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, Advances in Cryptology - CRYPTO 2001, volume 2139 of Lecture Notes in Computer Science, pages 213-229. Springer-Verlag, 2001.

Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, Advances in Cryptology - ASIACRYPT 2001, volume 2248 of Lecture Notes in Computer Science, pages 514-532. Springer-Verlag, 2001.

D. Dean and A. Stubblefield. Using Client Puzzles to Protect TLS. In Proceedings of the 10th USENIX UNIX Security Symposium, August 2001.

T. Dierks and C. Allen. The TLS protocol version 1.0. RFC 2246, IETF. http://www.ietf.org/rfc/rfc2246.txt, January 1999.

Gerhard Frey, Michael Muller, and Hans-Georg Ruck The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. IEEE Transactions on Information Theory, 45(5):1717-1719, 1999.

Steven D. Gaibraith, Keith Harrison, and David Soldera. Implementing the Tate pairing. In Claus Fieker and David R. Kohel, editors, Proc. Algorithmic Number Theory, 5th International Symposium (ANTS-V), volume 2369 of Lecture Notes in Computer Science, pages 324-337. Springer-Verlag, 2002.

M. Jakobsson and A. Juels. Proofs of Work and Bread Pudding Protocols. In Proceedings of the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security, September 1999.

Antoine Joux. A one-round protocol for tripartite Diffie-Hellman. In Wieb Bosma, editor, Proc. Algorithmic Number Theory, 4th International Symposium (ANTS-IV), volume 1838 of Lecture Notes in Computer Science, pages 385-394. Springer-Verlag, 2000.

Antoine Joux. The Weil and Tate pairings as building blocks for public key cryptosystems. In Claus Fieker and David R. Kohel, editors, Proc. Algorithmic Number Theory, 5th International Symposium (ANTS-V), volume 2369 of Lecture Notes in Computer Science, pages 20-32. Springer-Verlag, 2002.

A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In Proceedings of the ISOC Symposium on Network and Distributed Systems Security (SNDSS), pages 151-165, February 1999.

T. Okamoto and D. Pointcheval. REACT: rapid enhanced-security asymmetric cryptosystem transform. In Bart Preneel, editor, Topics in Cryptology - CT-RSA 2002, volume 2271 of Lecture Notes in Computer Science, pages 159-175. Springer-Verlag, 2002.

Eric R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In Birgit Pfizmann, editor, Advances in Cryptology - EUROCRYPT 2001, volume 2045 of Lecture Notes in Computer Science, pages 195-210. Springer-Verlag, 2001.

X. Wang and M. K. Reiter. Defending Against Denial-of-Service Attacks with Puzzle Auctions (Extended Abstract). In Proceedings of the IEEE Symposium on Security and Privacy, May 2003.

Appendix

[0067] Several details of the bilinear mapping used in the scheme for generating decryption-oriented puzzles in accordance with the invention are now discussed. In accordance with various embodiments of the invention, G_1 is chosen to be a large subgroup of the group of points on an elliptic curve over F_q of the order $l \neq q$, where $q = p^n$. G_2 is chosen to be a subgroup of $F^*_{q^r}$, where r is the security multiplier and $q^r - 1$ is divisible by l. Two different pairings can be defined over an elliptic curve, the Weil pairing and the Tate pairing. Because it is generally faster to compute, preference is generally given in accordance with the invention to the Tate pairing. In addition, a low security multiplier is needed for the pairing to be efficiently computed, and r will generally always reach its optimal value in the Tate pairing, but does not always do so for the Weil pairing.

[0068] Before defining the Tate pairing, some definitions are first provided. First, let $k(x,y)$ denote the field of quotients (i.e., the rational functions in x, y with coefficients in the field of k), where $f = g/h$ and both g and h are homogeneous of the same degree. Let the function field of the curve E, denoted k(E), be the equivalence classes of rational functions on E, $k(E) = k(x,y)/I(E)$. If $f = g/h \in k(E)$, then $h \notin I(E)$, and the two functions g/h and g'/h' are identified if $gh' = g'h$.

[0069] Let $k[E] = k[x,y]/I(E)$ be the coordinate ring of E (whose quotient field is $k(E)$ from before). Given $f \in k[E]$, the order of f at point p is denoted $\mathrm{ord}_P(f)$, and is the multiplicity of the point. Using $\mathrm{ord}(f/g) = \mathrm{ord}_P(f) - \mathrm{ord}_P(g)$, ord_P can be extended to $k(E)$. Given $f \in k(E)$, f can be said to have a zero at $P \in E$ if $\mathrm{ord}_P(f) > 0$, and a pole at P if $\mathrm{ord}_P(f) < 0$. Also, f has a zero at P if $f(P) = 0$, and a pole at P if $f(P)$ is not finite, denoted by $f(P) = \infty$.

[0070] The divisor group of a curve E, denoted Div(E), is the free Abelian group generated by the points of E. Thus, a divisor $D \in \mathrm{Div}(E)$ is a formal sum, $\sum_{P \in E} n^P (P)$, with $n_P \in Z$ and $n_P = 0$ for all but finitely many $P \in E$. The degree of D is defined by $\deg D = \sum_{P \in E} n^P$. The support of D is the set of points for which $n_P \neq 0$.

[0071] Given $f \in k(E)$, a divisor $\mathrm{div}(f)$ can be associated to f, given by $\mathrm{div}(f) = \sum_{P \in E} \mathrm{ord}^P (f)(P)$. A divisor D is principal if it has the form $D = \mathrm{div}(f)$ for some $f \in k(E)$ (f is unique up to constant multiples). Two divisors D_1 and D_2 are linearly

equivalent, denoted $D_1 \sim D_2$ if $D_1 - D_2$ is principal. Given an elliptic curve E and $D \in \mathrm{Div}(E)$, D is principal if and only if $\deg D = 0$ and $\sum n^P (P) = 0$. Given a function $f \in \mathrm{k}(E)$ and a divisor $D = \sum n^P (P)$, f can be evaluated at D by defining $f(D) = \prod f(\mathrm{P})^{n_P}$ for P in the support of D.

[0072] Returning to the Tate pairing, given $l \in Z$, $l \neq 0$, the l-torsion subgroup of E, denoted $E[l]$, is the set of points of order l in E. $\mathrm{E}[l] = \{P \in E: lP = O\}$. Given an l-torsion point P, D_P denotes a divisor from the class $(P) - (O)$ of the quotient of group of divisors of degree 0 by the subgroup of principal divisors, and f_P denotes a function such that $\mathrm{div}(f_P) = l(P) - l(O)$. The Tate Pairing of two points $P, Q \in E[l]$ is defined as $t(p, Q) = f_P(D_Q)^{(q^r - 1)/l}$. Recall that $l \mid q^r - 1$.

[0073] An elliptic curve defined over a field of characteristic q is supersingular if $E[q^r] = 0$ for all $r \geq 1$. Supersingular curves have extra endomorphisms in their endomorphism ring, and these endomorphisms map points defined over the ground field to points defined over an extension field. Thus, given an endomorphism ϕ, $t(P, \phi(P)) \neq 1$, there is no concern over the points in the pairing being linearly independent. Moreover, $t(P, \phi(Q))$ is denoted by $\hat{t}(P,Q)$.

[0074] Regarding hashing, recall that the signature generation scheme requires a cryptographic hash function I: $\{0,1\}^n \rightarrow G_1$. To construct such a hash function, a hash function from $\{0,1\}^n$ to F_q is first constructed, and then an encoding function from F_q to G_1 is constructed. Given $M \in \{0,1\}^n$, M is hashed to $f \in \{0,1\}^{\log q}$ and it is rehashed if it is not less than q. Let y be the f-th element of F_q. Given y, the encoding function calculates x from the equation for E, and sets $P = (x,y)$. The encoding function outputs lP which is an element in G_1.

[0075] As noted above, the present system can use the Weil pairing, as well as a pairing over more general Abelian varieties. More general bilinear maps of the form m: $G_0 \times G_1 \rightarrow G_2$ can also be used. In this case, both the ciphertext and signature can be shortened in length by taking G_0 to be a subgroup of F_p and G_1 to be a different subgroup of F_{p^s} of the same order. Both the Weil and the Tate pairings can be used on the asymmetric pair $G_0 \times G_1$ as the map of m.

What is claimed is:

1. A method for solving a problem using network devices in a computer network, the method comprising:

receiving, by a first network device, a first problem provided by a second network device;

providing a second problem, by the first network device, to a third network device, wherein the second problem is based at least in part on the first problem;

receiving a solution to the second problem by the first network device; and

solving the first problem, by the first network device, using the received solution to the second problem.

2. The method of claim 1, further comprising:

receiving a request from the second network device to use a resource of the first network device;

determining whether the solution to the second problem is valid; and

permitting the second network device to use the resource of the first network device if the solution to the second problem has been determined to be valid.

3. The method of claim 1, further comprising:

receiving a request from the second network device to establish a connection with the first network device;

determining whether the solution to the second problem is valid; and

permitting the second network device to establish a connection with the first network device if the solution to the second problem has been determined to be valid.

4. The method of claim 1, further comprising:

receiving a protocol request from the second network device by the first network device;

determining whether the solution to the second problem is valid; and

performing an operation described by the protocol request, by the first network device, if the solution to the second problem has been determined to be valid.

5. The method of claim 1, wherein the first network device is a server and the second and third network devices are clients.

6. The method of claim 1, wherein solving the second problem by the third network device requires using one or more computation resources of the third network device.

7. The method of claim 1, wherein the complexity of the second problem is varied based at least in part on the level of protection against resource-depletion attacks that is desired by the first network device.

8. The method of claim 1, wherein the first problem is a decryption-oriented puzzle that comprises a plurality of ciphertext portions.

9. The method of claim 8, wherein the decryption-oriented puzzle comprises two ciphertext portions, and wherein the solution to the second problem is obtained by solving one of the two ciphertext portions of the first problem.

10. The method of claim 9, wherein both of the two ciphertext portions are required to solve the decryption-oriented puzzle.

11. The method of claim 1, wherein the third computing device uses a private key associated with the first computing device to solve the second problem.

12. The method of claim 1, wherein the first computing device uses a private key and the solution to the second problem in solving the first problem.

13. The method of claim 1, further comprising solving the second problem by the first computing device when the solution to the second problem received by the first network device is invalid.

14. The method of claim 1, further comprising:

receiving, by the first network device, a third problem provided by the third network device;

providing a fourth problem, by the first network device, to the second network device, wherein the fourth problem is based at least in part on the third problem;

receiving a solution to the fourth problem by the first network device; and

solving the third problem, by the first network device, using the solution to the fourth problem.

15. The method of claim 1, further comprising establishing a secure cryptographic key, by the first network device, using the solution to the first problem.

16. A first network device in a computer network that receives a first problem from a second network device, that provides a second problem that is based at least in part on the

first problem to a third network device, that receives a solution to the second problem, and that solves the first problem using the received solution to the second problem.

17. An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for solving a problem, the computer readable program code means in the article of manufacture comprising:

computer readable program code means for causing a first network device to receive a first problem from a second network device;

computer readable program code means for causing the first network device to provide a second problem to a third network device, wherein the second problem is based at least in part on the first problem;

computer readable program code means for causing the solution to the second problem to be received by the first network device; and

computer readable program code means for causing the first network device to use the received solution to the second problem to solve the first problem.

18. A computer system for solving a problem, comprising:

means for receiving, by a first network device, a first problem provided by a second network device;

means for providing a second problem, by the first network device, to a third network device, wherein the second problem is based at least in part on the first problem;

means for receiving a solution to the second problem by the first network device; and

means for solving the first problem, by the first network device, using the received solution to the second problem.

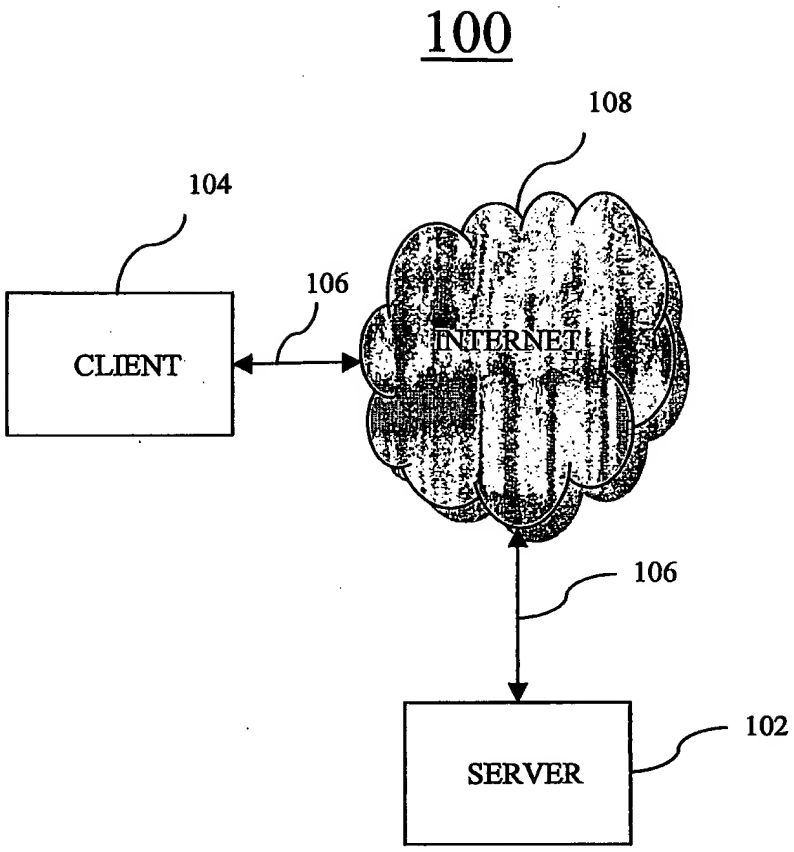


FIG. 1

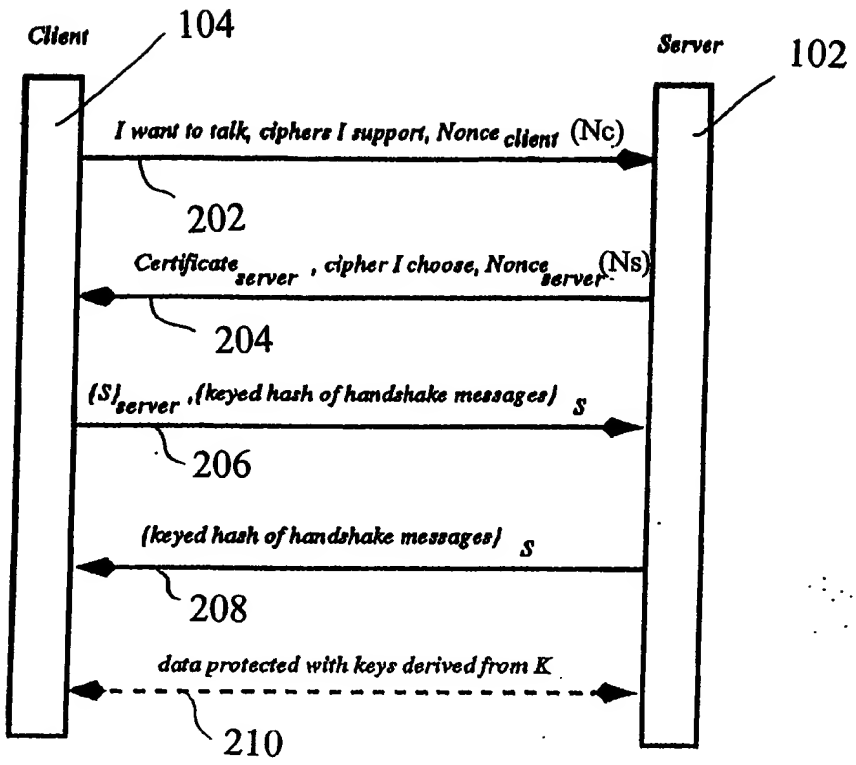


FIG. 2

(Prior Art)

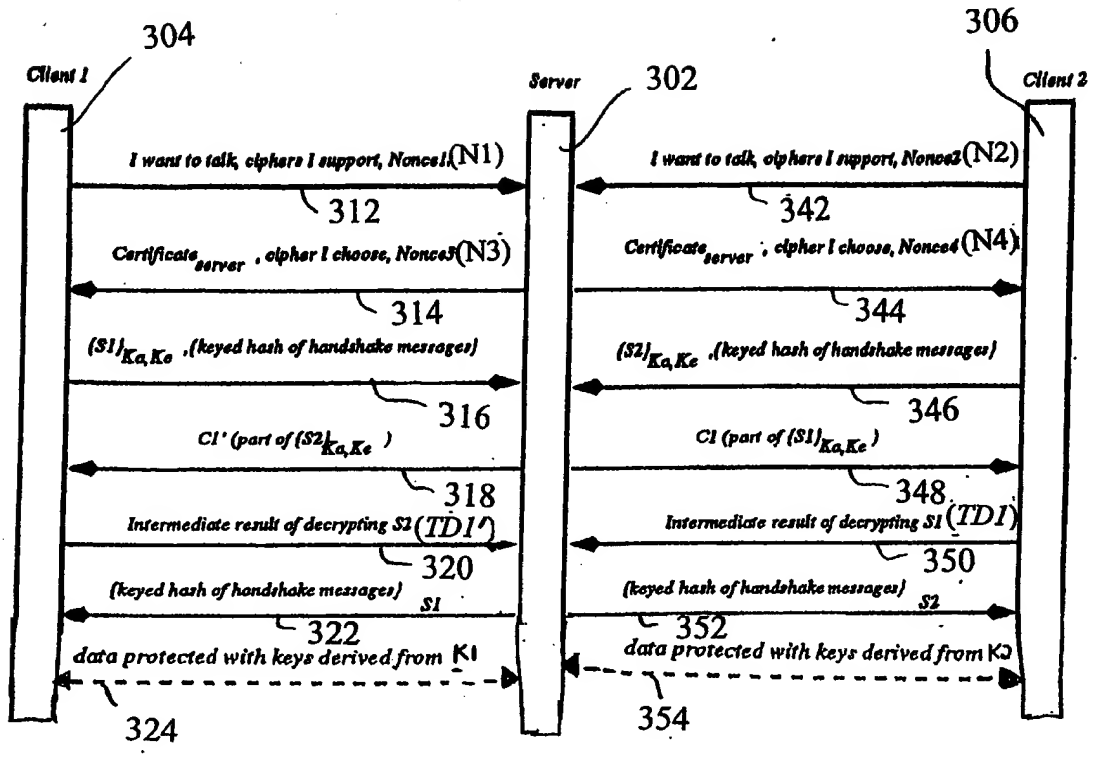

304
306
Client 1
Server
302
Client 2
I want to talk, ciphers I support, Nonce1 (N1)
312
I want to talk, ciphers I support, Nonce2 (N2)
342
Certificate server , cipher I choose, Nonce3 (N3)
314
Certificate server , cipher I choose, Nonce4 (N4)
344
(S1)Ka,Ke ,(keyed hash of handshake messages)
316
(S2)Ka,Ke ,(keyed hash of handshake messages)
346
C1' (part of (S2)Ka,Ke)
318
C1 (part of (S1)Ka,Ke)
348
Intermediate result of decrypting S2 (TD1')
320
Intermediate result of decrypting S1 (TD1)
350
(keyed hash of handshake messages) S1
322
(keyed hash of handshake messages) S2
352
data protected with keys derived from K1
324
data protected with keys derived from K2
354

FIG. 3

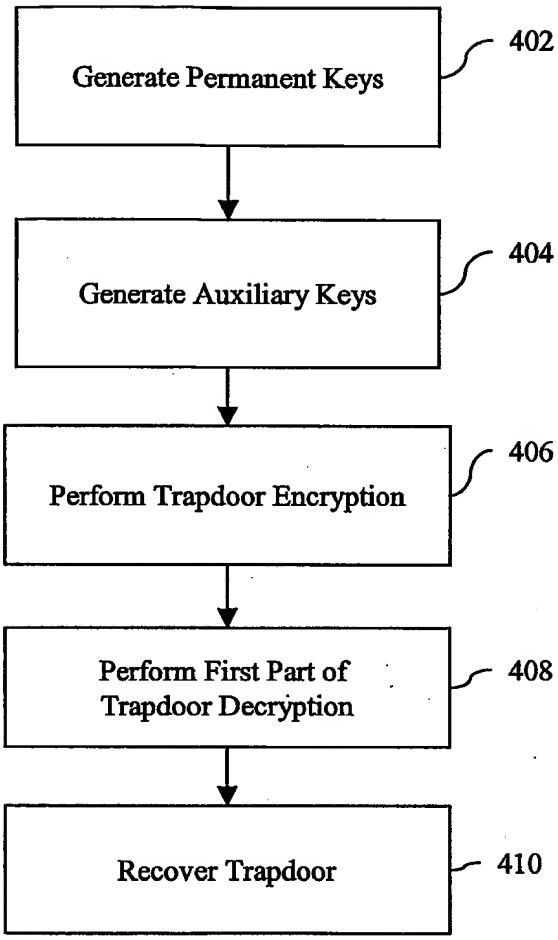

Generate Permanent Keys
402
Generate Auxiliary Keys
404
Perform Trapdoor Encryption
406
Perform First Part of Trapdoor Decryption
408
Recover Trapdoor
410

FIG. 4

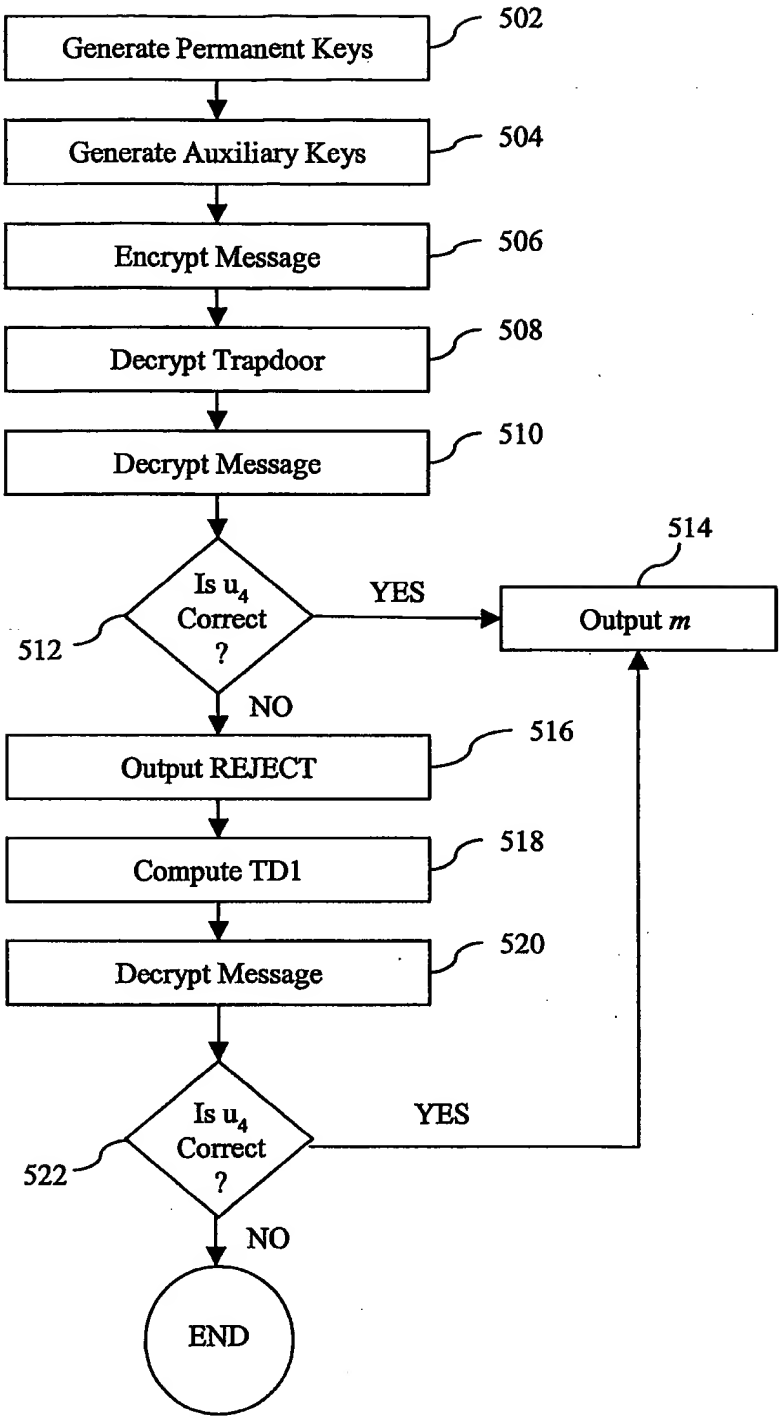

Generate Permanent Keys
502
Generate Auxiliary Keys
504
Encrypt Message
506
Decrypt Trapdoor
508
Decrypt Message
510
Is u_4 Correct ?
512
YES
Output *m*
514
NO
Output REJECT
516
Compute TD1
518
Decrypt Message
520
Is u_4 Correct ?
522
YES
NO
END

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US05/06245

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : H04L 9/00, 9/32;
US CL : 713/168, 170, 201;
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 713/168, 170, 201; 709/219

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Please See Continuation Sheet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	Diament et al., The Dual Receiver Cryptosystem and Its Applications, ACM CCS' 2004, October 2004, pp. 330-343 especially Figure 2, and Page 338 Section 5.4	1-18
A	US 2004/0030932 A1 (Juels et al.); 12 February 2004 (12.02.04); Fig. 5 and Paragraphs 0076 - 0086	1-18
A	US 6,192,473 B1 (Ryan, Jr. et al.) 20 February 2001 (20.02.01), Fig. 2A-2C and Col. 4 Line 36 - Col. 5 Line 30	1-18
A	T. Aura et al. DOS-resistant authentication with client puzzles. Security Protocols Workshop 2000, LNCS, 2133, 2000.	1-18
A	Juels et al. Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks. Proceedings of the ISOC Symposium on Network and Distriputed Systems Security, Pages 151-165, February 1999	1-18
A	Dean et al. Using Client Puzzles to Protect TLS. Proceedings of the 10th USENIX UNIX Security Symposium August 2001.	1-18

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
07 June 2005 (07.06.2005)

Date of mailing of the international search report
21 JUN 2005

Name and mailing address of the ISA/US
Mail Stop PCT, Attn: ISA/US
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
Facsimile No. (703) 305-3230

Authorized officer
Andrew Caldwell for Jessica R. Matthews
Telephone No. (571) 272-3868

Form PCT/ISA/210 (second sheet) (January 2004)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US05/06245

Continuation of B. FIELDS SEARCHED Item 3:
ACM; IEEE; Google
search terms: client, puzzle, authentication, denial, service